

ExactLearner: a Tool for Exact Learning of \mathcal{EL} Ontologies

Ricardo Duarte, Boris Konev, Ana Ozaki

Technische Universität Dresden, Germany

University of Liverpool, United Kingdom

KRDB Research Centre, Free University of Bozen-Bolzano, Italy

Abstract

We present **ExactLearner**, a tool for exactly learning and teaching \mathcal{EL} terminologies. The learning protocol follows Angluin’s exact learning model, where an ontology engineer tries to identify an ontology by interacting with a domain expert by asking queries. We implement the learning process as a question-answer game between two components of our system, the learner and the teacher. We evaluate **ExactLearner**’s performance on \mathcal{EL} ontologies from the Oxford ontology repository and demonstrate that despite the algorithm being exponential, it successfully terminates for small and medium size ontologies. We investigate the impact of various learner and teacher features and identify those most useful for learning.

Introduction

Authoring ontologies is a laborious task that requires a combined expertise of domain experts, who know the vocabulary of terms used in a particular subject area and have an understanding of the conceptual relationships between them, and of knowledge engineers, who can formalise these relations in an appropriate ontology definition language. In (Konev et al. 2018) the dialogue between an expert and a knowledge engineer is formalised as an instance of Angluin’s exact learning framework in which a *learner* tries to exactly identify an ontology by asking queries to a teacher, seen as an *oracle*. It is assumed that the vocabulary of terms is known and is communicated directly to the learner; in contrast, the exact ontology composition has to be found through a ‘trial and error’ learning process.

The learner poses queries of two kinds: membership queries, which ask the oracle to determine whether a given inclusion is entailed by the ontology, and equivalence queries, whether the ontology constructed is complete. If the answer to an equivalence query is negative, the oracle returns a statement which follows from the expert’s knowledge but not from the ontology constructed so far, or the other way around. As the domain expert may not be able to formulate ontologies in a clear

and succinct way, we make no assumptions about the statements returned by the teacher.

We are interested in algorithms that can identify any target ontology independently of the behaviour of the teacher. For complexity bounds, we consider the worst possible, adversarial teacher, which chooses to reveal as little information as possible.

The contributions of this paper are twofold. First, we build on results of (Konev et al. 2018) and give an algorithm for learning \mathcal{EL} terminologies, which is exponential in the size of concept expressions and its vocabulary but not in the size of the whole terminology. This result complements previous results showing that there is no polynomial time algorithm which can exactly learn (even acyclic) \mathcal{EL} terminologies (Konev et al. 2018). We then introduce **ExactLearner**, a tool for exactly learning and teaching \mathcal{EL} terminologies, which contains an implementation of our learning algorithm as well as a teacher. We evaluate **ExactLearner**’s performance on \mathcal{EL} ontologies from the Oxford ontology repository (Oxford) and demonstrate that despite the algorithm being exponential, it successfully terminates for small and medium size ontologies. We investigate the impact of various learner and teacher features and identify those most useful for learning. The missing proofs can be found in the full version of this paper available at <https://exactlearner.github.io>.

Related work. Most relevant to our work are: the DL-Learner (Lehmann 2009), which learns concept expressions (but not ontologies) in various fragments of description logic using refinement operators; and systems based on the exact learning model such as: Logan-H (Arias, Khardon, and Maloberti 2007) for learning function-free first order Horn sentences from interpretations; and EIRENE (Alexe et al. 2011) for learning schema mappings. For a more detailed discussion of related work, see (Konev et al. 2018).

Preliminaries

Description logic. Let N_C and N_R be countably infinite sets of *concept* and *role* names. An \mathcal{EL} concept expression C is formed according to the rule: $C, D := A \mid \top \mid C \sqcap D \mid \exists r.C$, where A ranges over

N_C and r ranges over N_R . A (general) \mathcal{EL} *concept inclusion* has the form $C \sqsubseteq D$, where C and D are \mathcal{EL} concept expressions. An \mathcal{EL} *ontology* is a finite set of \mathcal{EL} concept inclusions (Baader, Brandt, and Lutz 2005). We call an \mathcal{EL} ontology \mathcal{O} a *terminology* if for all $C \sqsubseteq D \in \mathcal{O}$ either C or D is a concept name and \mathcal{O} has at most one¹ inclusion of the form $A \sqsubseteq C$ for every $A \in N_C$. \mathcal{EL}_{lhs} is the class of \mathcal{EL} terminologies consisting only of inclusions of the form $C \sqsubseteq A$, while \mathcal{EL}_{rhs} only of inclusions of the form $A \sqsubseteq C$.

The *size* $|C|$ of a concept expression C is the length of the string that represents it, where concept and role names are considered to be of length one. The *vocabulary* $\Sigma_{\mathcal{O}}$ of an ontology \mathcal{O} is the set of concept and role names occurring in \mathcal{O} . The size of a concept inclusion $C \sqsubseteq D$, denoted $|C \sqsubseteq D|$, is $|C| + |D|$ and the size of an ontology \mathcal{O} , denoted $|\mathcal{O}|$, is $\sum_{C \sqsubseteq D \in \mathcal{O}} |C \sqsubseteq D|$. The semantics of \mathcal{EL} is defined as usual (Baader et al. 2003). We write $\mathcal{I} \models \alpha$ to say that a concept inclusion α is true in \mathcal{I} . An interpretation \mathcal{I} is a *model* of an ontology \mathcal{O} if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{O}$. $\mathcal{O} \models \alpha$ means that $\mathcal{I} \models \alpha$ for all models \mathcal{I} of \mathcal{O} ; and $\mathcal{O} \equiv \mathcal{O}'$ means that $\mathcal{O} \models \alpha$ if and only if $\mathcal{O}' \models \alpha$ for all concept inclusions α .

Subsumption learning framework. Given a class of ontologies \mathcal{L} (for example all ontologies in a particular DL, \mathcal{EL} terminologies etc), we are interested in the exact identification of a *target ontology* $\mathcal{O} \in \mathcal{L}$ by posing queries to an oracle. We assume that the vocabulary of the target terminology $\Sigma_{\mathcal{O}}$ is known to the learner. A *membership query* is a call to the oracle to test for an inclusion $C \sqsubseteq D$, where C, D are $\Sigma_{\mathcal{O}}$ -concept expressions of the DL under consideration, if $\mathcal{O} \models C \sqsubseteq D$. An inclusion $C \sqsubseteq D$ is a *positive example* w.r.t. a target \mathcal{O} if $\mathcal{O} \models C \sqsubseteq D$ and a *negative example* else. An *equivalence query* is a call to the oracle to check if a *hypothesis* ontology \mathcal{H} is equivalent to the target \mathcal{O} . If it is the case, the oracle responds ‘yes’, otherwise the oracle returns a positive example $C \sqsubseteq D$ with $\mathcal{H} \not\models C \sqsubseteq D$ or a negative example $E \sqsubseteq F$ with $\mathcal{H} \models E \sqsubseteq F$. Such a positive example $C \sqsubseteq D$ (negative example $E \sqsubseteq F$) is called a *positive counterexample* (a *negative counterexample*, resp.) to \mathcal{H} being equivalent to \mathcal{O} . For a formal definition of the *subsumption learning framework* and a discussion of how this definition relates to Angluin’s exact learning model see (Konev et al. 2018).

We say that a class of ontologies \mathcal{L} is *exactly learnable* if there is an algorithm, which halts for any target $\mathcal{O} \in \mathcal{L}$ and computes, using membership and equivalence queries, $\mathcal{H} \in \mathcal{L}$ with $\mathcal{H} \equiv \mathcal{O}$. An ontology class is exactly learnable in polynomial time if it is exactly learnable by an algorithm A such that at every step² of computation the time used by A up to that step is bounded by

¹In the literature, the term *terminology* commonly refers to sets of concept inclusions $A \sqsubseteq C$ and concept definitions $A \equiv C$, with no concept name occurring more than once on the left. As $A \equiv C$ can be equivalently rewritten as $A \sqsubseteq C$ and $C \sqsubseteq A$, our definition is a natural extension of this one.

²We count each call to an oracle as one step.

Algorithm 1 The learning algorithm for \mathcal{EL}

Input: An \mathcal{EL} terminology \mathcal{O} given to the oracle; $\Sigma_{\mathcal{O}}$ given to the learner
Output: An \mathcal{EL} terminology \mathcal{H} computed by the learner such that $\mathcal{O} \equiv \mathcal{H}$

- 1: Set $\mathcal{H} = \{A \sqsubseteq B \mid \mathcal{O} \models A \sqsubseteq B, A, B \in \Sigma_{\mathcal{O}}\}$
- 2: **while** $\mathcal{H} \neq \mathcal{O}$ **do**
- 3: Let $C \sqsubseteq D$ be the returned positive counterexample for \mathcal{O} relative to \mathcal{H}
- 4: Compute $C' \sqsubseteq D'$ with C' or D' in $\Sigma_{\mathcal{O}} \cap N_C$
- 5: **if** $C' \in \Sigma_{\mathcal{O}} \cap N_C$ **then**
- 6: Compute a right \mathcal{O} -essential α from $C' \sqsubseteq D' \sqcap \prod_{C' \sqsubseteq F' \in \mathcal{H}} F'$
- 7: **else**
- 8: Compute a left \mathcal{O} -essential α from $C' \sqsubseteq D'$
- 9: **end if**
- 10: Add α to \mathcal{H}
- 11: **end while**
- 12: **return** \mathcal{H}

a polynomial $p(|\mathcal{O}|, |C \sqsubseteq D|)$, where \mathcal{O} is the target and $C \sqsubseteq D$ is the largest counterexample seen so far. \mathcal{EL}_{lhs} and \mathcal{EL}_{rhs} are known to be exactly learnable in polynomial time, while the class of all \mathcal{EL} ontologies is not learnable in polynomial time (Konev et al. 2018).

Learning \mathcal{EL} Ontologies

In this section we present Algorithm 1, which can exactly learn \mathcal{EL} terminologies in time exponential in $|\mathcal{O}|$, the size of the largest concept expression in \mathcal{O} , and $|\Sigma_{\mathcal{O}}|$, the size of the ontology vocabulary, but not in the size of the whole ontology.

In the main loop of the algorithm the learner poses an equivalence query to the oracle. If the oracle answers “yes” then the algorithm returns \mathcal{H} equivalent to \mathcal{O} . Otherwise, it receives a counterexample $C \sqsubseteq D$. It is easy to see that at all times $\mathcal{O} \models \mathcal{H}$ so the counterexample is always positive.

As \mathcal{O} is a terminology, complex C and D in the counterexample can only “connect” via a concept name, which can be identified by asking membership queries. This is formalised by the following lemma proved by the canonical model construction.

Lemma 1. *Given a positive counterexample $C \sqsubseteq D$, one can construct, by posing membership queries, a positive counterexample $C' \sqsubseteq D'$ such that $|C' \sqsubseteq D'| \leq |C \sqsubseteq D|$ and either C' or D' is a concept name in time polynomial in $|\mathcal{H}|$, $|C|$ and $|\Sigma_{\mathcal{O}}|$.*

Having transformed the counterexample to the case of a concept name on the left or on the right, the algorithm tries to minimise the size of the counterexample. If C' is a concept name then Algorithm 1 merges D' with the right-hand sides of all inclusions in \mathcal{H} with C' on the left (if they exist) and computes a so called *right \mathcal{O} -essential* counterexample. Otherwise, D' is a concept name, and the algorithm computes a *left \mathcal{O} -essential*

counterexample. It then adds the resulting \mathcal{O} -essential concept inclusion α to \mathcal{H} .

To explain the left and right \mathcal{O} -essential counterexamples, following (Konev et al. 2018), we identify in the obvious way each \mathcal{EL} concept expression C with a finite tree T_C whose nodes are labelled with sets of concept names and whose edges are labelled with roles.

Right \mathcal{O} -essential concept inclusion α is computed by applying exhaustively the following rules to $A \sqsubseteq C$:

Concept saturation for \mathcal{O} : If $\mathcal{O} \models A \sqsubseteq C'$ and C' results from C by adding a concept name A' to the label of some node, then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

Sibling merging for \mathcal{O} : If $\mathcal{O} \models A \sqsubseteq C'$ and C' is the result of identifying in C two r -successors of the same node then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

Decomposition on the right for \mathcal{O} : If d' is an r -successor of d in C , A' is in the node label of d , and $\mathcal{O} \models A' \sqsubseteq \exists r.C_{d'}$ plus $A' \not\sqsubseteq_{\mathcal{O}} A$ if d is the root of C , then replace $A \sqsubseteq C$ by

- (a) $A' \sqsubseteq \exists r.C_{d'}$ if $\mathcal{H} \not\models A' \sqsubseteq \exists r.C_{d'}$; or
- (b) $A \sqsubseteq C|_{d\downarrow}$, otherwise, where

C_d is the concept corresponding to the subtree rooted in d and $C|_{d\downarrow}$ is the concept corresponding to the result of removing the subtree rooted in d from C .

We illustrate the transformation rules with examples.

1. For $\mathcal{H} = \emptyset$ and $\mathcal{O} = \{\text{Human} \sqsubseteq \exists \text{hasParent.Human}\}$ the oracle can return an arbitrary long `hasParent` chain starting at `Human` as a counterexample, for instance, `Human` \sqsubseteq `$\exists \text{hasParent}.\exists \text{hasParent}.\top$` is a chain of length two. With concept saturation, this counterexample can be strengthened to `Human` \sqsubseteq `$\exists \text{hasParent}.\text{Human} \sqcap \exists \text{hasParent.Human}$` , which is equivalent to \mathcal{O} .
2. For $\mathcal{O} = \{\text{Human} \sqsubseteq \exists \text{hasParent}.\text{Human} \sqcap \text{Male}\}$ and $\mathcal{H} = \{\text{Human} \sqsubseteq \exists \text{hasParent.Human}\}$, upon receiving a counterexample `Human` \sqsubseteq `$\exists \text{hasParent.Male}$` , the learner merges its right hand side with the right hand side of the inclusion in \mathcal{H} to form `Human` \sqsubseteq `$\exists \text{hasParent.Male} \sqcap \exists \text{hasParent.Human}$` and then strengthens it by sibling merging to form the inclusion in \mathcal{O} .
3. For $\mathcal{H} = \emptyset$ and $\mathcal{O} = \{\text{Woman} \sqsubseteq \text{Human}, \text{Human} \sqsubseteq \exists \text{hasParent.Human}\}$, even with concept saturation, there exist infinitely many chain counterexamples; `Woman` \sqsubseteq `$\text{Human} \sqcap \exists \text{hasParent}.\text{Human}$` is one of them. This inclusion can be decomposed at the root into (a) `Human` \sqsubseteq `Woman` and (b) `Human` \sqsubseteq `$\exists \text{hasParent}.\text{Human}$` . Picking either of them allows the learner make progress.

Left \mathcal{O} -essential concept inclusion α is computed by applying exhaustively the following rules to $C \sqsubseteq A$.

Concept saturation for \mathcal{H} : If $\mathcal{H} \models C \sqsubseteq C'$ and C' results from C by adding a concept name A' to the label of some node, then replace $C \sqsubseteq A$ by $C' \sqsubseteq A$.

Decomposition on the left for \mathcal{O} : If d is a non-root node such that $\mathcal{O} \models C|_{d\downarrow} \sqsubseteq A'$ and $\mathcal{H} \not\models C|_{d\downarrow} \sqsubseteq A'$, for some $A' \in \Sigma_{\mathcal{O}}$, then replace $C \sqsubseteq A$ by $C|_{d\downarrow} \sqsubseteq A'$; if $\mathcal{O} \models C_d \sqsubseteq A'$ and $\mathcal{H} \not\models C_d \sqsubseteq A'$, for some $A' \in \Sigma_{\mathcal{O}}$, then replace $C \sqsubseteq A$ by $C_d \sqsubseteq A'$.

The applicability of a rule may depend on the application of another rule. For example, for $\mathcal{H} = \{\exists \text{hasParent}.\top \sqsubseteq \text{Human}\}$ and $\mathcal{O} = \mathcal{H} \cup \{\exists \text{hasChild.Human} \sqsubseteq \text{Human}\}$ a counterexample could be `$\exists \text{hasChild}.\exists \text{hasParent}.\top \sqsubseteq \text{Human}$` , which can only be decomposed on the left for \mathcal{O} if we apply concept saturation for \mathcal{H} first.

Our proof of termination of Algorithm 1 and its complexity bound is based on the following lemma. To simplify the presentation we use $\sharp_{\mathcal{O}}$ to denote $|C_{\mathcal{O}}| \cdot |\Sigma_{\mathcal{O}}| + 1$.

Lemma 2. *Given a positive counterexample $C \sqsubseteq D$ for \mathcal{O} relative to \mathcal{H} , one can construct a positive counterexample $C' \sqsubseteq D'$ such that $|C' \sqsubseteq D'| \leq \sharp_{\mathcal{O}}$ in polynomial time in $|C \sqsubseteq D|$, $|\Sigma_{\mathcal{O}}|$ and $|\mathcal{H}|$.*

Since there are at most $|\Sigma_{\mathcal{O}}|^{\sharp_{\mathcal{O}}}$ many inclusions over $\Sigma_{\mathcal{O}}$ of size $\sharp_{\mathcal{O}}$, at most $|\Sigma_{\mathcal{O}}|^{\sharp_{\mathcal{O}}}$ counterexamples get added to \mathcal{H} over the run of the algorithm. Thus we obtain the following theorem.

Theorem 1. *The class of \mathcal{EL} terminologies is exactly learnable by Algorithm 1 in $O(|\Sigma_{\mathcal{O}}|^{2|C_{\mathcal{O}}| \cdot |\Sigma_{\mathcal{O}}| + 2} \cdot (|C \sqsubseteq D|)^2)$ time, where $C_{\mathcal{O}}$ is largest concept expression in \mathcal{O} and $C \sqsubseteq D$ is the largest counterexample seen so far by the algorithm.*

Concept saturation, sibling merging and decomposition on the right are all essential—hence the name—steps of the polynomial learning algorithm for $\text{DL-Lite}_{\mathcal{R}}^{\exists}$, which extends \mathcal{EL}_{rhs} with inverse roles and role hierarchies (Konev et al. 2018). Indeed, Algorithm 1 polynomially learns \mathcal{EL}_{rhs} .

Theorem 2. *The class of \mathcal{EL}_{rhs} terminologies is exactly learnable in polynomial time by Algorithm 1.*

Evaluation

We have implemented our learning algorithm in the `ExactLearner` system, available at <https://github.com/ExactLearner/ExactLearner>, in Java using the OWL API (Horridge and Bechhofer 2011) and the ELK reasoner (Kazakov, Krötzsch, and Simancik 2014). `ExactLearner` has two main components: a learner and a teacher.

The learner supports (1) “Concept Saturation”, (2) “Sibling Merging”, (3) “Decomposition”, applied on the right side of inclusions, and (4) “Concept Desaturation”, (5) “Sibling Branching” and (6) “Decomposition”, applied on the left. Operations (1), (2), (3) and (6) have already been described. In addition, we have also implemented (4) and (5), which act as heuristics to construct smaller, more informative counterexamples. Concept desaturation tries to remove concept names from nodes in the left of counterexamples to make them logically stronger. Sibling branching tries to strengthen a counterexample by splitting paths on the left. For example,

	p	# timeouts	avg CE	avg max C
Test 2:	0.01	3	17.2	27.7
	0.5	25	107.8	26.6
	1.0	26	190.4	19.5
Test 3:	0.01	2	5.6	31.7
	0.5	3	6.1	31.6
	1.0	3	6.3	31.9

Table 1: Learner against the adversarial teacher.

for $\mathcal{O} = \{\exists \text{hasDegree.BSc} \sqcap \exists \text{hasDegree.MSc} \sqsubseteq \text{PG}\}$ and $\mathcal{H} = \emptyset$, the inclusion $\exists \text{hasDegree.}(\text{BSc} \sqcap \text{MSc} \sqcap \text{PhD}) \sqsubseteq \text{PG}$ is a counterexample, from which desaturation removes the irrelevant PhD and then sibling branching strengthens it to the one in \mathcal{O} .

We have evaluated ExactLearner’s performance on \mathcal{EL} ontologies from the Oxford ontology repository (Oxford). Out of 797 ontologies in the repository, 174 (when ignoring object and data properties) are in \mathcal{EL} ; all but one are \mathcal{EL} terminologies. As a first experiment we ran the learner against a naïve teacher, which presents the target ontology inclusions one by one without modification. This experiment aims at estimating the overheads of the learning process under the best possible conditions. In this first experiment, for 50 out of 174 \mathcal{EL} terminologies computations concluded within 1 hour.

We selected these ontologies for further experiments. The selected ontologies range in size from 9 to 11 177 inclusions with vocabulary sizes ranging from 23 to 9334 concept names and from 2 to 25 role names. The average size of counterexamples produced by the teacher was 5.48 while the average size of the largest concept in \mathcal{O} was 2.7. The average size of the largest concept in \mathcal{H} was 31.3, an increase caused by concept saturation on the right side of inclusions. The performance bottlenecks in our system are for checking if the presented inclusion is a counterexample w.r.t. the current hypothesis ontology at the teacher side and entailment checks performed by the learner.

To challenge the learner, we have introduced an adversarial teacher, which forces the learner to apply particular operations from (1)–(6) above by manipulating the counterexamples. For instance, to force the learner to perform concept saturation on the right of $A \sqsubseteq C$, the teacher exhaustively tries to remove concept names from every node in the tree representation of C , while ensuring that the modified inclusion is still a counterexample. All in all, the adversarial teacher can apply: (7) “Concept Desaturation” on the right, which we have just described; (8) “Sibling Branching” on the right, which weakens counterexamples of the form $A \sqsubseteq \exists r.(C \sqcap D)$ into $A \sqsubseteq \exists r.C \sqcap \exists r.D$ (provided the latter is still a counterexample); (9) “Concept Saturation” on the left; and (10) “Sibling Merging” on the left, which are the opposite of learner’s concept desaturation and sibling branching. We also substitute concept definitions into

counterexamples, for instance, if $A \sqsubseteq \exists r.B$ is a counterexample and $B \sqsubseteq C \in \mathcal{T}$ we test $A \sqsubseteq \exists r.C$ for being a counterexample as well. We call this operation (11) “Composition on the right”. (12) “Composition on the left” is its counterpart. Operations (7)–(12) are applied at random with set probabilities so that the level of difficulty could be controlled.

Table 1 presents statistics of running the learner against the adversarial teacher. In Test 2 the teacher was applying transformations (7)–(12) with probability p of 0.01, 0.5 and 1.0. The learner can cope with a small distortion of examples ($p = 0.01$) but a significant distortion leads to a big increase in the number of time-outs. Figure 1 shows the percentage of the rules applied by the learner in Test 2 for $p = 0.01$.

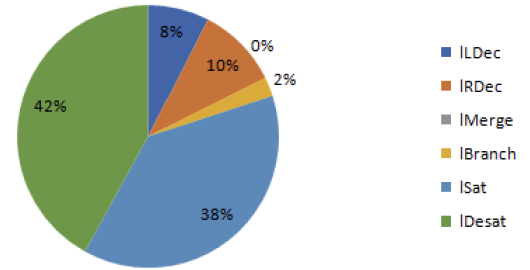


Figure 1: Usage of rules (1)–(6) by the learner when the oracle applies rules with probability 0.01.

As Figure 1 indicates, the most frequently applied rule (42%) is desaturation on the left. Its frequency grows to 94% when $p = 0.5$ and to 96% when $p = 1.0$. For the oracle, the most frequent (99%) rule, when $p = 0.01$, is saturation on the left. However, the frequency of composition on the left jumps from less than 1% to 90% when $p = 0.5$ and to 92% when $p = 1.0$. This change can be explained by the growth in the *absolute* number of applications of saturation on the left leading to an increase in the number of concept names available for composition. The discrepancy between the number of compositions on the left by the oracle and decompositions on the left by the learner is due to the fact that the oracle applies the rule repeatedly while the learner finds a minimal subtree in one rule application.

In Test 3 we have disabled rule (9), which leads to expensive saturation-desaturation, as well as rules (8) and (10) as the latter two rules almost never applied, yet took up a significant time in our tests. This has led to a significant drop in the failure rate even though other adversarial teacher operations were applied with a high probability. The high amount of applications of rule (9) performed by the teacher and of rule (4) performed by the learner suggests that the main cause of time-outs is the exponential explosion in the size of the vocabulary rather than the size of concepts in the target ontology.

We also measured the increase on the number of queries in Test 2 when the probability for the oracle to apply a certain rule increases. In Test 2, 22 ontology com-

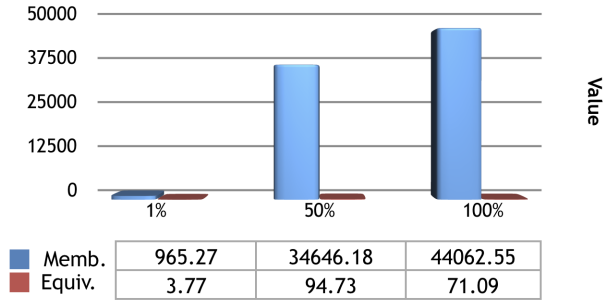


Figure 2: Membership and equivalence queries.

putations concluded within 1 hour with the probability of the oracle to apply a certain rule set to 1.0 (the oracle always apply all rules exhaustively). Figure 2 shows the increases in the average numbers of queries of both types asked by the learner when the probabilities for the oracle to apply its rules are set to 0.01, 0.5 and 1.0, compared to the case when no counterexample transformation rules are applied by the oracle (the baseline values for the number of membership and equivalence queries when no rules are applied are 6230148.55 and 41.23, respectively). The number of membership queries visibly increases as probabilities increase, while, in comparison, the number of equivalence queries remains nearly the same. This is expected, since computing \mathcal{O} -essential inclusions from less informative counterexamples need more membership queries. Though, since the learner indeed computes \mathcal{O} -essential counterexamples, there is only a small impact on the number of equivalence queries.

Playing with the Teacher

Our prototype teacher component can also be accessed via a graphical interface allowing a user to play the game of learning an ontology by posing as few membership and equivalence queries as possible.

Figure 3 presents a screenshot of the game after 2 queries. The bar on top of the ‘Equivalence query’ button allows the player to adjust the difficulty: a higher difficulty means that the probability for the oracle to apply its transformation rules is higher.

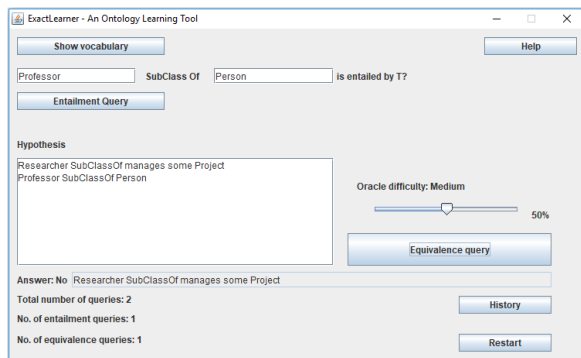


Figure 3: Learning game after 2 queries.

Conclusion

We presented ExactLearner, a prototype tool for exactly learning, and teaching, \mathcal{EL} ontologies. We demonstrated its applicability to small and medium size ontologies. We identified the size of the ontology vocabulary as the main cause of the performance bottleneck.

As future work, we plan to extend our algorithm to an ontology-based data access setting (Konev, Ozaki, and Wolter 2016) and adopt the Probably Approximately Correct (PAC) learning model extended with membership queries, so that our algorithm can also run without the teacher. We also plan to investigate the complexity of exactly learning of \mathcal{EL} terminologies in the PAC learning setting under different probability distributions.

Acknowledgements. We would like to thank Frank Wolter for fruitful discussions and Liyi Zhao for her contribution to an earlier version of ExactLearner. Konev was supported by the EPSRC project EP/H043594/1. Ozaki was supported by the PROVDL project and by the Center for Advancing Electronics Dresden (cfaed).

References

- Alexe, B.; ten Cate, B.; Kolaitis, P. G.; and Tan, W. C. 2011. EIRENE: interactive design and refinement of schema mappings via data examples. *PVLDB* 4(12):1414–1417.
- Arias, M.; Khardon, R.; and Maloberti, J. 2007. Learning horn expressions with LOGAN-H. *Journal of Machine Learning Research* 8:549–587.
- Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. 2003. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press.
- Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the \mathcal{EL} envelope. In *IJCAI*, 364–369. Professional Book Center.
- Horridge, M., and Bechhofer, S. 2011. The OWL API: A java API for OWL ontologies. *Semant. web* 2(1):11–21.
- Kazakov, Y.; Krötzsch, M.; and Simancik, F. 2014. The incredible ELK - from polynomial procedures to efficient reasoning with EL ontologies. *J. Autom. Reasoning* 53(1):1–61.
- Konev, B.; Lutz, C.; Ozaki, A.; and Wolter, F. 2018. Exact learning of lightweight description logic ontologies. *Journal of Machine Learning Research* 18(201):1–63.
- Konev, B.; Ozaki, A.; and Wolter, F. 2016. A model for learning description logic ontologies based on exact learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 1008–1015.
- Lehmann, J. 2009. DL-Learner: Learning concepts in description logics. *Journal of Machine Learning Research* 10:2639–2642.
- Oxford. Information systems group ontologies. Retrieved from <https://www.cs.ox.ac.uk/isg/ontologies/>. Accessed: 18 May 2018.